Express Mail Label EL493861947US
Docket No. 40611/117974

COMPREHENSIVE SECURITY STRUCTURE PLATFORM

FOR NETWORK MANAGERS

Inventors: Rajeev Khanolkar
Ozakil Azim
Rishi Asthana
Niten Ved
Kevin Hanrahan
Amit Ghildiyal
Shirisha Pogaku
Dhani Amaratunge
K.V. Rao Samavenkata

CLAIM TO PRIORITY

This application claims the benefit of U.S. Provisional Application No. 60/213,967 filed on June 23, 2000, and U.S. Provisional Application No. 60/219,650 filed on July 21, 2000, the applications and their disclosures being incorporated herein by reference in their entirety for all purposes.

FIELD OF THE INVENTION

This invention relates generally to computer network security and more particularly to monitoring computer networks for network security events.

BACKGROUND OF THE INVENTION

With a trend toward ever larger computer communications networks, especially Internet-based networks, the number of access points for potential intruders in a given system likely increases. Password attacks, spoofing, network scanning and sniffing, denial of service (i.e., any activity preventing the normal operation of network resources), and TCP/IP (Transmission Control Protocol/Internet Protocol) attacks are only a few of the types of damaging intrusion techniques to which a network may be subject. To safeguard against attack, intrusion, and other security threats, network systems in a typical Internet infrastructure may include intrusion detection systems, firewalls, virtual private networks (VPN's), web servers, anti-virus servers, email servers, authentication (AAA) servers, proxy servers, and network vulnerability assessment devices, among other servers and devices. Because these systems themselves interact with sources outside the network, they also provide access points for an attack or intrusion upon a network.

Logging is the procedure by which operating systems record events in the system as they happen. Within the logging memory of these devices, and other devices such as web servers, e-mail servers, DNS servers, etc., logs are kept that contain data comprising information chronicling network intrusion events. Presented with log data, however, monitoring devices often fail in two respects. First, they fail to effectively monitor log data from all relevant components on the network. Second, they fail to record and report the log data in a form that is timely and useful to network administrators. Moreover, while various systems such as firewalls and intrusion detection systems, such as NetRanger from Cisco Systems, Inc., may issue real time alarms to a network administrator of an intrusion event based on log data, within a network such alarms may be lost in the midst of numerous notices of intrusion events received by a network administrator. What is needed is a system to process and organize network intrusion events and log data from a number of network systems and provide them to a user in an interface that summarizes them, yet has links to more detailed information, that provides for real time notice and communications regarding current events, and that allows for the compilation and recalling of past log data and intrusion events for detection of patterns of activity for later use and consultation.

SUMMARY OF THE INVENTION

An invention that satisfies those needs and provides other benefits that will be apparent to one skilled in the art has now been developed. Broadly, in one aspect the present invention concerns a security monitoring system for computer networks to analyze and report on network intrusion events taking place on network service devices using their log data. The word "intrusion" should be broadly understood to include any type of security breach and accidental or inadvertent misuse as well as an actual intrusion. Thus, it is to be understood that "network intrusion event," or "intrusion event," covers any type of network security event.

In response to an intrusion event, the security monitoring system can issue intrusion alarms to network administrative users ("users"), who will then be able to obtain information regarding intrusion events in real time on a display screen. The system filters log data, which contains information related to intrusion events, to provide a more manageable flow of data that can be more easily reviewed by a system administrator because the data relating to intrusion events are not "lost" in large amounts of noise (e.g., data not relating to intrusion events). The system has means for organizing and collating intrusion event data within a searchable database accessible to a user through a reporting system that can generate security reports and summaries of intrusion events for network service devices and that provides information in response to user queries.

The system has discrete software modules that receive and process log data from various network devices. Using Java programming language as a foundation, and utilizing relational database management systems (RDMS), log data chronicling network activity events received from various network devices are converted to event objects for processing and manipulation by the system. Event objects may contain information on the source device type, data and time of an intrusion event, host name (i.e., the particular device), alarm identification of the intrusion event from the network device, source Internet Protocol ("IP") address of the network device, source port of the network device, destination IP address (foreign IP address of intrusion source), destination port (port of intrusion source), protocol monitored (e.g., TCP, UDP, ICMP), and the intrusion event itself as recorded by that device. The user may set filters that regulate the type and amount of log data received, limiting what passes the filter to only particular sources, particular event types, and/or particular protocols.

Each event object that is created is read, and the intrusion event information it contains is assigned a severity level. Event objects meeting or exceeding a predetermined threshold severity level, or other threshold criteria, may be broadcast to the user and displayed as an intrusion alarm on a user interface display screen in real time. Users may set filters regulating the stream of event objects received as broadcasts based on severity level or other criteria or may choose to receive all event objects regardless of severity or other criteria as broadcast intrusion alarms.

The user interface display screen displays broadcast intrusion alarms and provides access to the corresponding event object information via an alarm console. The alarm console is connected to a broadcast subsystem that includes modules enabling the user to connect to other users in a "chat" connection to inform other users in the network of real time intrusion alarms. From the report console, the user may receive summary reports of network security status and of event objects contained in the database. Using relational database connectivity systems, the database collates and organizes event objects received, allowing them to be recalled according to user queries input into the report console. The report console is linked to the database through a report subsystem, which provides a report servlet to access and transmit data for display from the database.

Another aspect of the invention is a distributed network of monitoring systems, each accessible to a central console for viewing broadcast intrusion alarms remote from the source computer system. The central console can also remotely search the individual databases of each monitoring system in the distributed network.

Another aspect of the invention is a method for detecting and monitoring network intrusion events from log data received from network service devices in a computer network using the system of this invention.

Other features and advantages of the present invention will be apparent to one skilled in the art from the following detailed description and drawings. As used herein, "in communication with" and "coupled" include direct and indirect (i.e., through an intermediate) communication and coupling.

BRIEF DESCRIPTION OF THE DRAWINGS

To facilitate further discussion of the invention, the following drawings are provided.

Figure 1 is a block diagram showing an overview of the computer system of one embodiment of the invention.

Figure 2 is a block diagram of the architecture of the event handling subsystem.

Figures 3a, 3b, and 3c contain a table illustrating network device intrusion events and assigned severity levels used by that computer system.

Figure 4 is a flow chart depicting the operation of the event handling subsystem of that computer system.

Figure 5 is a block diagram of an alternative embodiment of a portion of the event handling subsystem.

Figure 6 is a block diagram of a distributed network of the computer system of Figures 1-4.

Figure 7 is a block diagram of the client side architecture of the computer system of Figures 1-4.

Figure 8 is a representation of the user display screen of the computer system of Figures 1-4.

Figure 9 is a flow chart of the operation of the query function of the computer system of Figures 1-4.

Figure 10 is a block diagram of the user display screen of the computer system of Figures 1-4.

DETAILED DESCRIPTION OF THE INVENTION

In Figures 1 and 2, the security structure platform system of the present invention, hereafter system 10, comprises event handling subsystem 50, syslog listener 51, syslog manager 53, reporting agent 52, one or more event parsers 54, event manager 55, event broadcaster 56, event saver 57, database 58, web client subsystem 60, web client interface 30, alarm console 32, report console 34, reporting subsystem 40, report servlet 45, application reporter 48, broadcast subsystem 70, chat server 75, report 26, query 28, alarm 22, and chat 24.

In Figure 1, system 10 receives log data 18 from multiple network service device sources, which sources may include firewalls, VPN (Virtual Private Network) routers/servers, e-mail servers, authentication servers, and other network devices that are accessible from sources outside the network, such as independent Internet Protocol ("IP") sources. For purposes of example, Figure 1 identifies log data 18 from Intrusion Detection Server ("IDS") 11, Firewall 12, VPN Router/Server 13, E-mail server 14, Proxy Server 15, and "N" sources 16 ("N" represents other network devices). Although a plurality of different network service device sources are illustrated, the invention may monitor only one network service device or multiple network service devices of the same type, e.g., multiple firewalls. It will also be appreciated that other network service devices such as Web Servers, Anti-virus Servers, Calendar Servers, Directory Servers, DNS Servers, and Network Probes, among other devices in a computer network, may also provide log data to be processed.

System 10 is preferably a web-based platform, implemented on, for instance, Linux or Solaris server platforms, and is driven by web-interface browsers such as Netscape 4.x and Internet Explorer IE4.x. System 10 operates in conjunction with a web server, such as Apache or Netscape. Java Data Base Connectivity (JDBC) based connections are preferably used to retrieve data stored in various tables using a relational database management system ("RDMS"). The database (not pictured in Figure 1) itself uses proprietary software (for instance Oracle 8.0) for its implementation.

Incoming log data 18, containing network intrusion information from the network devices, such as, for instance, event type, source IP, date and time of event, and firewall connection information, are received and processed within event handling subsystem 50. Event handling subsystem 50 parses log data 18, converting them into event objects that contain information regarding details of an intrusion event rendered in a standard format for processing and collating. Based on user-customizable, pre-determined criteria, the event handling subsystem may determine that the event object is of a sufficient severity to generate an alarm 22 to client subsystem 60, where it is displayed on web client interface 30 at alarm console 32. Alarm console 32 has a chat 24 electronic communications link, allowing the user to connect with other users on-line in the network, e.g., to determine status of an alarm event or to notify others of an alarm and the need for remedial action. Web client interface 30 may be a graphic user interface on a web browser having a display screen displaying the screens for both alarm console 32 and report console 34.

The screen for report console 34 on web client interface 30 may be displayed alongside the screen for alarm console 32 so that both screens can be consulted simultaneously by a user, e.g., a network security administrator or other network administrator. Report console 34 displays summaries and reports concerning network intrusions and may monitor specific network devices, 11-16, and summarize reports therefrom. Using report console 34, which operatively interacts with reporting subsystem 40, the user may compose and issue queries 28 for status, reports, and history, e.g., the user may issue a query on the status of a network device such as firewall 12, or the user may search the database (not pictured) for archived event objects based on, for instance, IP source, if the administrator notes a developing pattern of intrusion. Reports 26 of results from the query can be displayed on report console 34. Web client interface 30, alarm console 32, and report console 34 may, in some embodiments, be accessible to any user with access to system 10, for instance from a web browser, allowing a plurality of people to access and communicate back and forth with system 10.

Figure 2 shows the application architecture of event handling subsystem 50. Modules of event handling subsystem 50 operate through threads launched by a software engine (not pictured) operating on the server platform. Data are received at a network port connection, which may be a 514/UDP (User Datagram Protocol) port. These threads run operations organized into classes generally identifying the operation to be performed. The system operates on a relational database format. Thus, program classes are organized and named to allow transfer and processing of data from other databases in the network devices. A common means of containing and transferring data, such as Java Beans, can provide the system of data exchange, and Java based Event Objects, in extended form, can provide the means for standardizing the log data for further processing in the system.

Inbound log data from the network devices 11-16 may be posted in a particular data format, such as syslog, extensible markup language ("xml"), or simple network management protocol ("snmp"), among others. Thus, in one embodiment, event handling subsystem 50 reads and processes syslog data and uses syslog daemons to forward data. Those reporting devices that post log data in syslog, for instance, firewall 12, forward log data directly to syslog listener 51 via a 514/UDP port (not pictured). Those devices that cannot post data via syslog have their log data processed by reporting agent 52, which reads the log and generates a syslog message reproducing the log lines read. Reporting agent 52 then sends the message to syslog listener 51. In one embodiment, event handling subsystem 50 is configured to generate multiple reporting agents 52 to read and forward messages from a plurality of reporting devices in the network, which devices may generate alternative formats for their log data. Alternatively, reporting agent 52 may have a multi-thread capability where each thread monitors a discrete reporting device.

Syslog listener 51 can be filtered according to the preference of the user. Filters may be activated through the web client interface 30 and may restrict receipt of log data based on, for instance, application name, host name, event severity, internal device alarm identifications, source address, destination address, destination port, and protocol.

After being received by syslog listener 51, log data containing syslog messages are detected, read, and serialized before streamed to syslog manager 53. Syslog manager 53 receives each message and matches the type of reporting device to the date, providing timestamps to the message and attaching informational strings. The syslog message is then streamed by syslog manager 53 to a specific event parser 54 configured to parse event objects from that particular type of reporting device. For instance, one parser may be configured to parse log data from an intrusion detection system such as NetRanger. Another parser may be configured for a Cisco PIX Firewall. Assignment to the correct event parser 54 may be done by matching the log data for the particular reporting device against application identifications present in the system among event parsers 54 stored within a class consulted by syslog manager 53. Using the application identification, a handle for the associated event parser for the particular device is retrieved. The syslog message is then streamed to the appropriate event parser 54.

While the foregoing embodiment illustrates operation of system 10 using syslog messages, it is to be understood that system 10 may also be configured to process messages that are in a format other than syslog, such as xml or snmp. Thus, modules such as syslog listener 51, reporting agent 52, and syslog manager 53, among other modules disclosed in the embodiment, may process messages in an alternative format.

Each event parser 54 contains threads for an abstract class launched for instances of log data from each reporting device of the type for which event parser 54 may be configured. Event parser 54 parses the syslog message to create an event object. In an extension of the Java.util Event Object class, the event object includes further information fields relevant to network security monitoring. For instance, once parsed, the created event object contains coded information, which may include the event type, application, reporting device, event time stamp, application time stamp, source IP of the event, destination IP of the event, and event duration, as well as any identification number that may be assigned by the reporting device to the event type. A user may also direct event parser 54 to filter out (i.e., reject) log data based on these fields of information, in which case event parser 54 will restrict receipt of filtered log data and not process them into an event object. As with syslog listener 51, event parser 54 filters can be set through web client interface 30.

Event parser 54 then streams the event object it has created to event manager 55. Event manager 55 processes the event object, evaluating it according to pre-determined criteria, which may be based on the type of the event, and assigns a severity level. Based on severity level, event manager 55 filters the event object, thereby determining accordingly whether the event object is to be broadcast and/or to be saved.

The levels of severity assigned may be as follows:

Alert Messages, Severity 1

Critical Messages, Severity 2

Error Messages, Severity 3

Warning Messages, Severity 4

Notification Messages, Severity 5

Informational Messages, Severity 6

Debugging Messages, Severity 7

Application of these severity levels for the Cisco PIX Firewall to certain event types is illustrated in Figures 3a, 3b, and 3c.

Severity filters within event manager 55 may be set by a user using configuration tables accessible through web client interface 30. Filtering within event manager 55 may also be based on the event type, i.e., certain event types would not be evaluated for severity level and/or broadcast. In event manager 55, as well as in other system modules and features, filter settings may be set by a user (for instance, a network administrator) through web client interface 30 (not pictured in Figure 2) and loaded by the software engine upon startup of system 10. Settings may be modified by a user during system 10 operation by further input into web client interface 30. The software engine may be notified whenever the filters are modified and the filters may be reloaded by the software engine from a database. Upon input completion, settings are modified at the appropriate module or feature (for instance, at event manager 55, syslog listener 51, and/or event parser 54) by the software engine. Consequently, where no filter is activated, all event objects received by event manager 55 will be broadcast, regardless of severity and/or event type. Where the filter is activated, event manager 55 can be directed not to forward for broadcast any event objects having, for example, a severity level of one. Filters may also be set depending on the identification numbers assigned to the particular event type by the specific network device. Identification numbers pertaining to event types may also be filtered at syslog listener 51 or event parser 54, i.e., prior to the creation of the event object.

As with the filters available for activation at syslog listener 51 and/or event parser 54, event manager 55 may filter based on other criteria and threshold levels that may be set by the user as a broadcast threshold, including source IP address, source port, destination IP address, and destination port. When these filters are in place, only those event objects having the permitted source ports, destination ports, or IP addresses will be broadcast to alarm console 32 for real time viewing. If an event object is to be streamed for broadcast (i.e., meets any applicable threshold level or criteria) event manager 55 streams the event object to event broadcaster 56, which may occupy a TCP port and listen for an event object stream thereon. Event objects may also be saved to database 58 by streaming them from event manager 55 to event saver 57.

One advantage of the present system over prior monitor systems is that event manager 55 receives the event object before it is stored in a database. Therefore, the determination of whether to broadcast the event object as an intrusion alarm is made nearly instantaneously upon receipt of the event object. That the event object need not be collected first, retrieved from a database, and only then examined to determine whether to broadcast an alarm allows the intrusion alarm to be broadcast in real time and without delay.

Event saver 57 is a thread launched for each event object to be saved to database 58. Using a class for database access, event saver 57 contacts database 58 and saves the event object. In database 58, the event object is archived and organized according to its particular information for later retrieval or matching with patterns or signatures. Database 58 may be one of many commercially available database systems, for example, Oracle 8.0. Event broadcaster 56 receives event objects to be broadcast to the web client interface 30 or more specifically to alarm console 32 of the interface. Event broadcaster 56 establishes and maintains a communications connection with alarm console 32, for instance, a TCP pipe, to stream the event object for display as an intrusion alarm.

In Figure 4, inbound log data from a device source are provided at function block 100. Functional block 110 represents conforming inbound log data 100 (for instance, in the exemplary embodiment log data messages posted in syslog would conform) and/or non-conforming log data retrieved by an agent 102 activated to read the log data and provide a syslog message format, or other conforming format, for them. At block 130 syslog listener receives the log data and directs them to one of a plurality of event parsers that is provided for the particular device source at block 140. At block 150 event parsers 140 read the log data and parse the information contained therein into a particular form, for instance, an event object, for further processing. At this stage, an assigned severity level based on event type and other information pertaining to an intrusion event may be included in the created event object. The event object proceeds to the event manager via decision block 160. If the event object is of a sufficiently high level of severity or meets other threshold criteria, it is forwarded as a generated alarm 170 to the alarm console for broadcast. The broadcast step 170 may be accomplished over a TCP pipe to the alarm console.

At step 160, event objects may be designated for broadcast as alarms based on threshold criteria other than severity, e.g., event objects pertaining to a specified event such as a cable failure, which may be based on filter settings applied at step 160 within the event manager or on filters 105, which may restrict receipt of event objects at step 160. Threshold severity levels, and other threshold criteria, may be set by a user or a default mode retained. It is also contemplated that a user may set no threshold criteria or threshold level and allow the generation of alarms and broadcast of all event objects received at 160.

Event objects of all severity levels, unless directed otherwise, are forwarded via block 180 to event saver for saving to the database (not pictured). Alarms received by the event broadcaster 170 are processed and sent as intrusion alarms to the alarm console. Filters 105, which can be set by a user, may restrict receipt of log data or broadcast of event objects at various stages of the system, based on certain criteria such as event type or severity level.

Figure 5 shows an alternative embodiment of the invention in which log data from security devices are received and parsed by a single event parser 76 embodied within an event aggregator 71. This embodiment uses a generic (standard) message format, e.g., extensible markup language (xml), thus eliminating the need for separate event parsers. Security devices that send data in xml are received directly by the event aggregator 71. Reporting agent 73 may be used for devices that cannot post data in an xml or other standard format (e.g., syslog format). The agent establishes a connection with the event aggregator 71, downloads the appropriate parser from it, parses the security event information in the log, and sends an xml representation of the security event to event aggregator 71 for filtering.

Event aggregator 71 comprises a generic event parser 76, which parses log data received and creates event objects. Filter 77 contained within the event aggregator filters security event objects based on event type and severity filtering criteria (among other programmable criteria). An aggregator 79 then aggregates the events, eliminating or combining redundant events where necessary to reduce volume, and forwards the created event objects to the event manager 55 for further processing.

In Figure 6 a plurality of network systems 10a, 10b, 10c,..., 10n (subscript "n" indicates the "nth" system) are in operation and in communication with a central operations interface (e.g., console) 30a in addition to local operations consoles 30b, 30c, ..., and 30n. From central operations console 30a, a user can access event objects stored in databases 58a, 58b, 58c, ..., and 58n of the various systems though the reporting subsystems 40b, 40c, ..., 40n as well as central reporting system 40a. Central operations console 10a can issue commands and queries to all other consoles 10b, 10c, ..., and 10n to be received by local users.

Figure 7 illustrates the client side of the system architecture. Web user interface 30 displays both alarm console 32 and report console 34. In one embodiment, consoles 32 and 34 are displayed on a single web browser screen that may be manipulated as necessary by the user according to the operations software used, e.g., MS Windows. In one embodiment, alarm console 32 is a Java applet program loaded on the client browser. Alarm console 32 is responsible for real time event display, real time device status display, alarm display, and chat communications. Alarm console 32 is connected to broadcast subsystem 70 and the modules constituting the subsystem. Chat server 75 establishes a communications link over secure sockets to enable web user interface 30 to link with other web user interfaces so as to communicate with other web client users, for instance regarding an alarm event.

Event broadcaster 56 forwards intrusion alarms to alarm console 32, along with accompanying event object information. Intrusion alarm event objects are displayed on a real time device status panel appearing on console 32 showing the status of devices configured for the user. An illustration of this screen appears as Figure 8. On web client interface 30, the user can set display filters restricting the event objects viewed on alarm console 32 without setting system wide filters (i.e., within the modules of the subsystems). For instance, users can filter out display of event objects from particular devices or filter out display of event objects pertaining to particular event types so that each user may focus on particular devices and/or events types in the network. Thus, a user may filter broadcast event objects without filtering out event objects received by event manager 55, thereby allowing unviewed event objects to be saved to database 58 or to be broadcast to other possible users on web client interface 30 accessing the same system. User settings for display filters may be stored in a database.

Color codes for the various devices, indicating levels of severity and other data information, may appear in a scrolled listing. Alarms appear on alarm console 32 through a Java applet window, scrolling as they are received in summary line form and in real time. These summary lines not only represent event objects identified for broadcast, but each summary line is also a hypertext link to further information contained in the event object. Selecting (or clicking) on a line allows the user to drill down to further graphic framesets revealing more information regarding the event.

Alarm console 32 also allows real time communications with other users through a connection to chat server 75, opening a java applet window where messages may be composed and sent to other operators on-line. Additionally, chat server 75 also shows which other users may also be on-line at the time and their names. This is illustrated in Figure 8 (see the Operator Chat Window).

As shown in Figure 7, report console 34 can access the reporting subsystem 40. Reporting subsystem 40 is accessed using a Hypertext Transfer Protocol ("HTTP") communication connection from the web browser in web user interface 30. Custom queries for event object searches may be input to reporting subsystem 40 by the user. These queries are received and processed by report servlet 45, which is responsible for returning results of the query after initiating a search of database 58 conducted through application reporter 48. Commonly used queries can also be input by selecting them from a query menu. The following table illustrates some sample queries from a query menu for the Cisco PIX Firewall application:

-List Messages for a Period

-Connection Query by Source and Destination IP

-Denied Outbound Connection Query by Source and Destination IP

-Denied Inbound Connection Query by Source and Destination IP

-Authorization Query by User

-Alarm ID Query

Figure 9 illustrates the processing of a query by report servlet 45. Upon initiation by querant 300, report servlet 45 opens a connection to application reporter (not pictured in Figure 9). An input query 310 from querant 300 is received and parsed (block 320) to identify various elements or criteria sought, such as application type, report name, or other criteria. The parsed elements are forwarded to the application reporter (block 330), which initiates a search of the database (block 340). A result set is generated from the database (block 350), which is returned to report servlet (block 360). Preferably, the database result set is in xml format. Report servlet 45 converts xml result data to hypertext markup language ("HTML") (block 370) and the results are returned to the querant 300 (arrow 380) for display on the report console.

Figure 10 illustrates one graphical user interface for use as the web client interface 30. Other configurations and formats will be apparent to those skilled in the art. Display screen 90 is separated into two sections, report console segment 91 and alarm console segment 92. Report console segment 91 features a main menu section 93, which displays the first two levels of the application in folder tree fashion, these being the application type, i.e., the type of device being monitored, and instances, i.e., the specific device. Submenu 94 provides a frame to display options for viewing the data where reports, summaries, and graphs may be displayed, as well as an area where queries and query results may be input. Optionally, user access to various filters in the system may be provided through Submenu 94, or filter access may be provided otherwise on the graphical user interface. This frame may also provide an area for requesting "help." Work area 95 can be used to list the various types of reports, summaries, and queries for each application type and display report summaries, query results, graphs, and online help. Alarm console segment 92 may be continuously displayed at the bottom of the screen with features allowing access to information concerning any security event or alarm issued by the system. Alarm console segment 92 may also be restored to work area 95 for ease of display.

Although the invention has been described in relation to specific embodiments, other variations and modifications will become apparent to those skilled in the art and the claims are intended to cover all embodiments falling within the true spirit and scope of the invention.